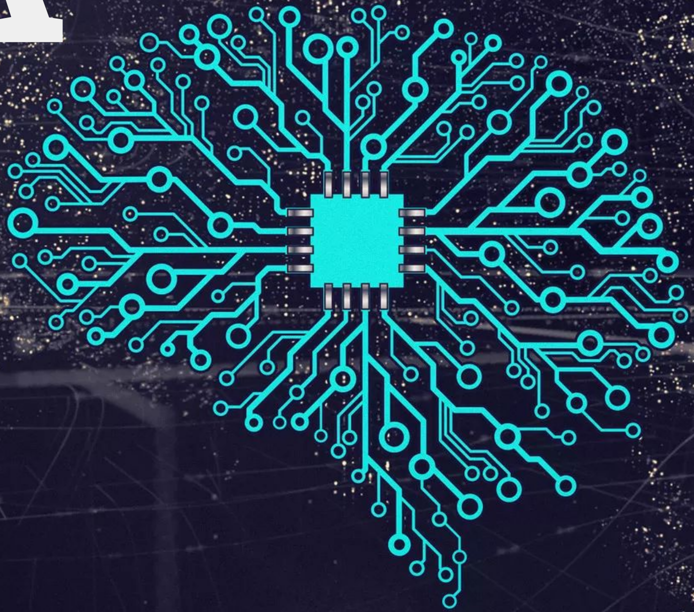
RNA

Redes Neurais Artificiais

*Aula 03



UNIVERSIDADE DO ESTADO DE MINAS GERAIS
UEMG
UNIDADE DIVINÓPOLIS

Docente: Felipe de Paula Oliveira
Graduação em Engenharia da Computação

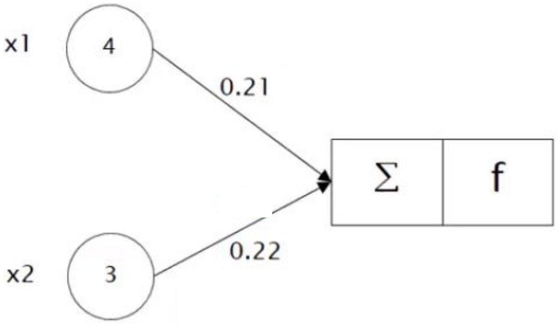
UNIVERSIDADE
DO ESTADO DE MINAS GERAIS
UEMG
UNIDADE DIVINÓPOLIS

ajuste de pesos

- **classificação**

x1 - comprimento do parafuso
x2 - diâmetro do parafuso
Classe A (0) e Classe B (1)

$$soma = \sum_{i=0}^{n} xi \; * wi$$



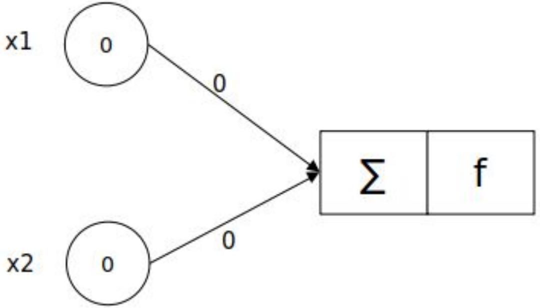
ajuste de pesos

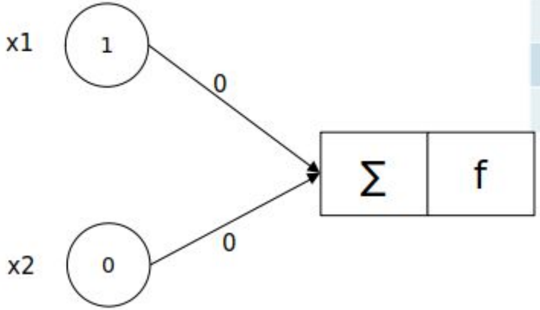
Operador E

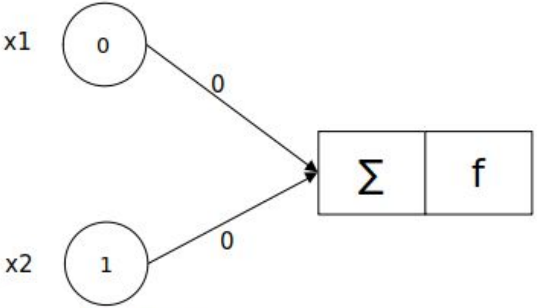
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

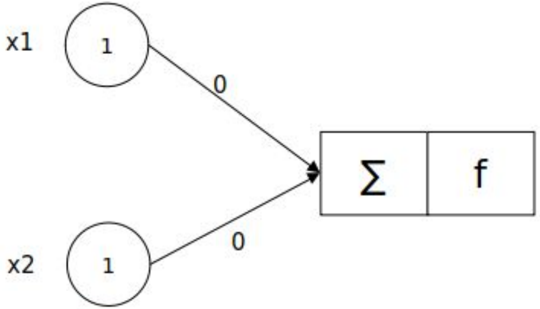
ajuste de pesos

x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1









ajuste de pesos

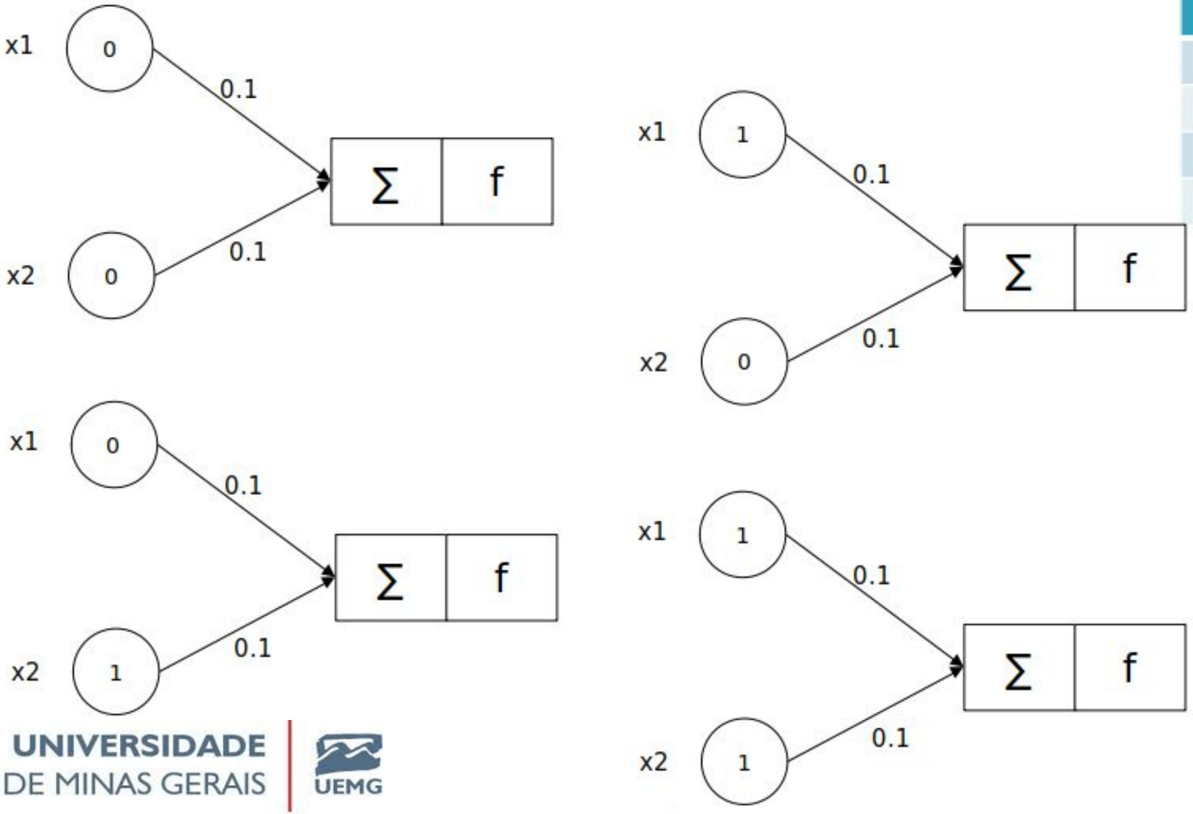
Algoritmo mais simples

- **erro = respostaCorreta – respostaCalculada**

Os pesos são atualizados até os erros serem pequenos

- **peso(n + 1) = peso(n) + (taxaAprendizagem * entrada * erro)**

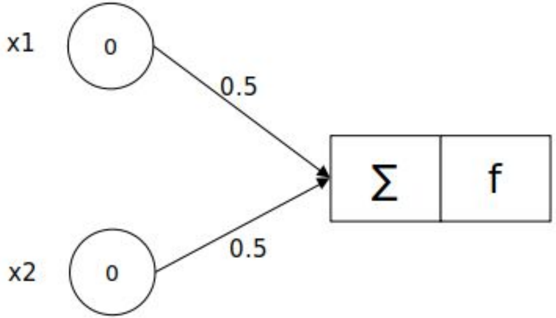
ajuste de pesos

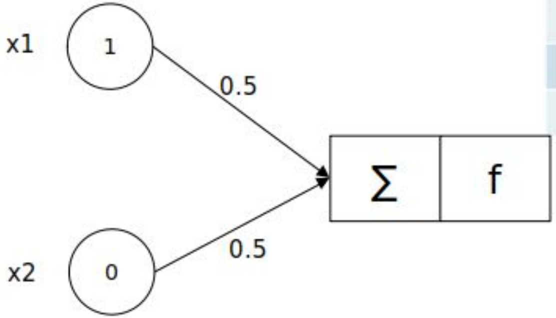


x1	x2	Class e
0	0	0
0	1	0
1	0	0
1	1	1

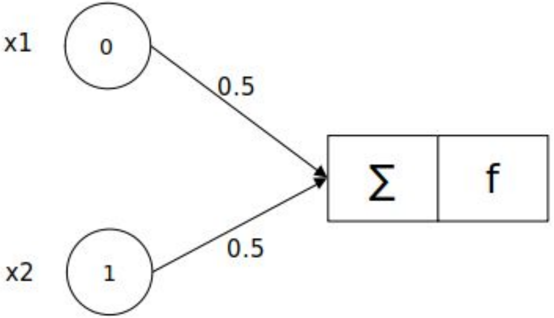
UNIVERSIDADE DO ESTADO DE MINAS GERAIS UEMG
UNIDADE DIVINÓPOLIS

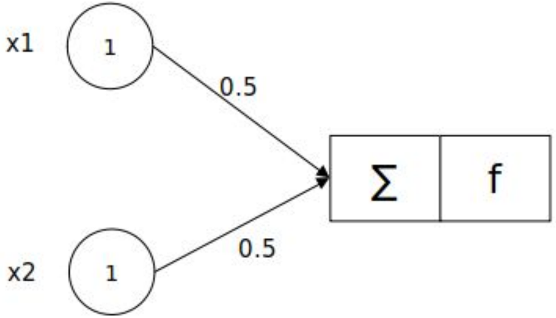
ajuste de pesos





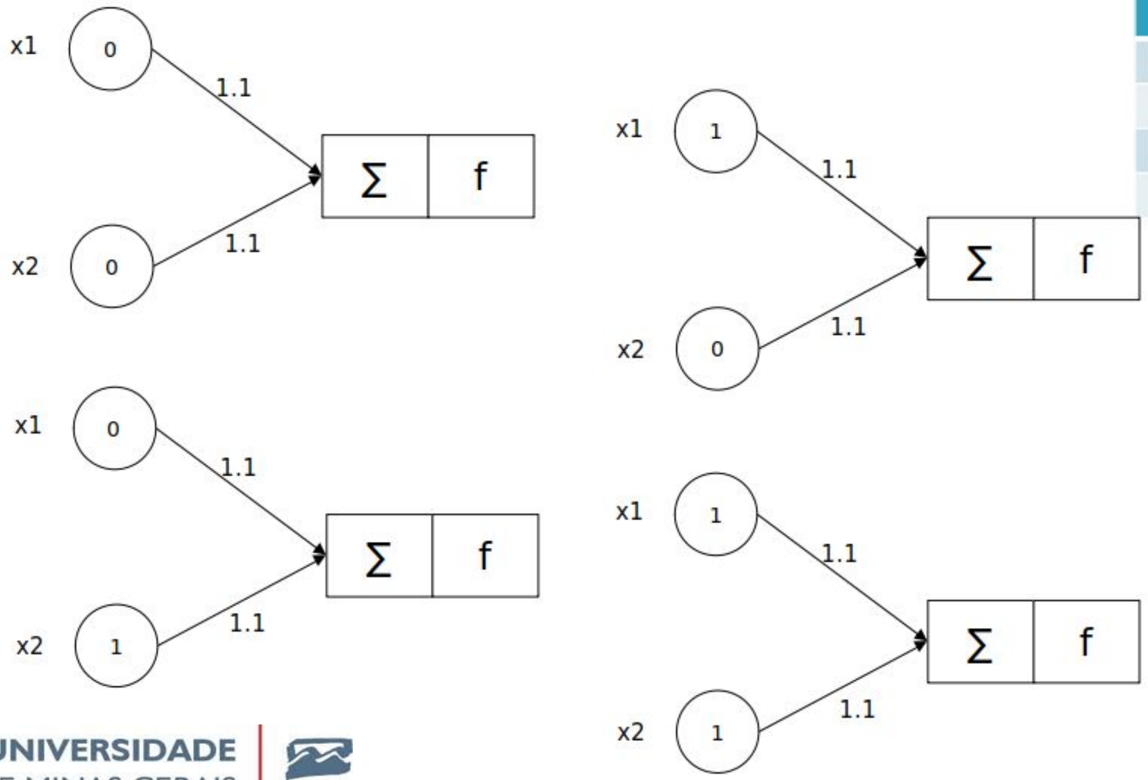
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



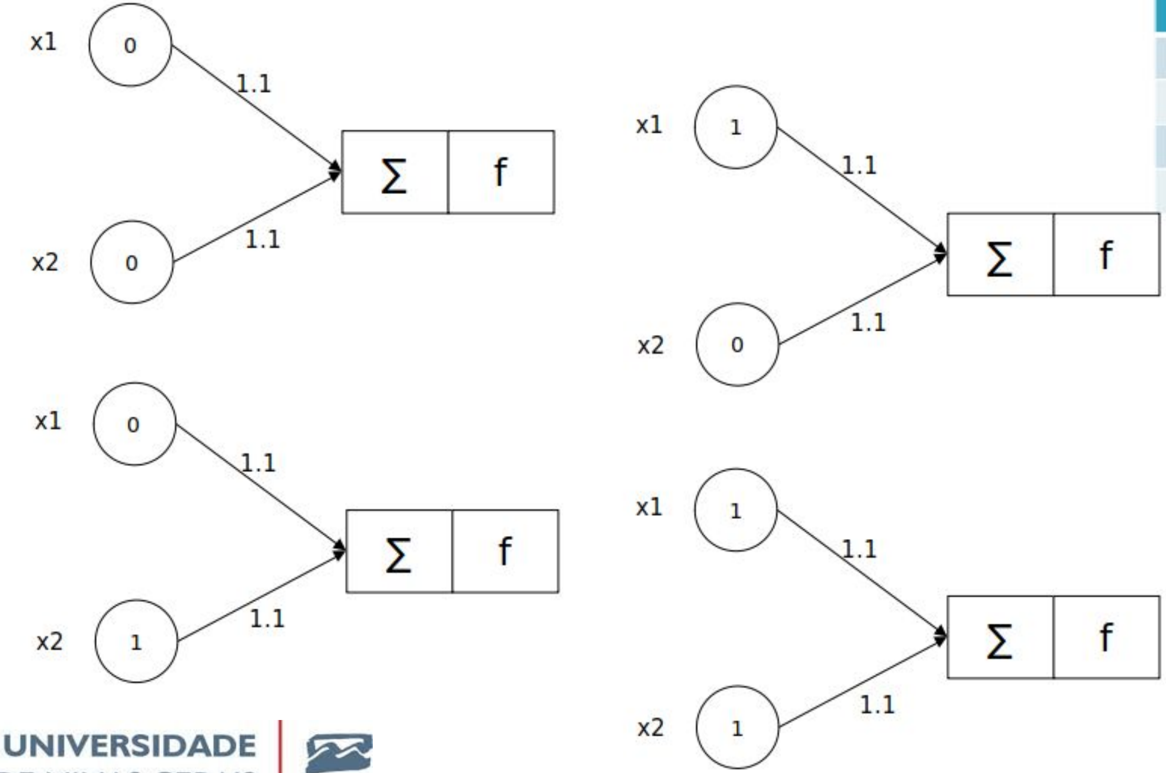


ajuste de pesos

x1	x2	Classe
0	0	0
0	1	1
1	0	1
1	1	1



ajuste de pesos



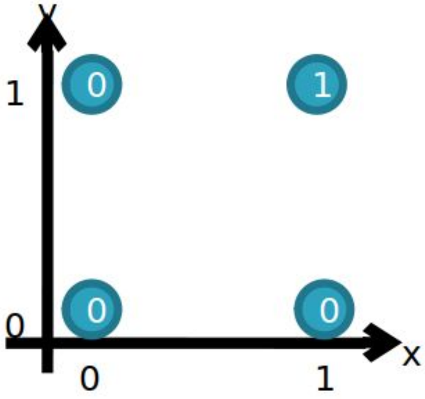
x1	x2	Classe
0	0	0
0	1	1
1	0	1
1	1	1

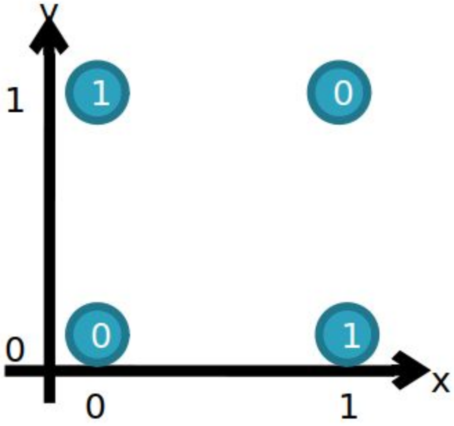
ajuste de pesos

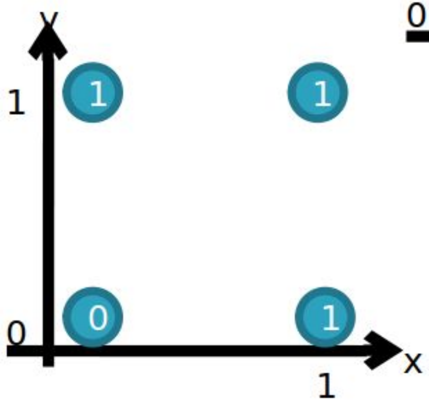
Enquanto o erro for diferente de zero

- **Para cada registro**
    - **Calcula a saída com os pesos atuais**
    - **Compara a saída esperada com a saída calculada, somando o erro**
    - **Para cada peso da rede**
        - **Atualiza o peso - peso(n + 1) = peso(n) + (taxaAprendizagem * entrada * erro)**

ajuste de pesos

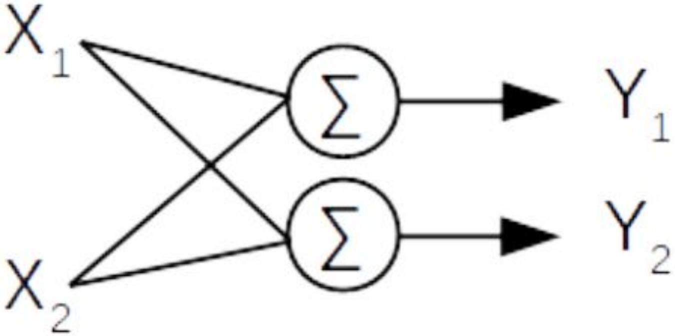


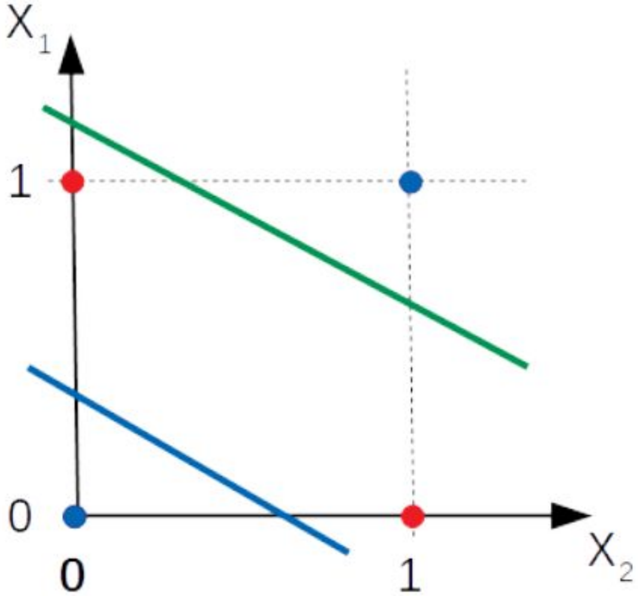




problemas e soluções

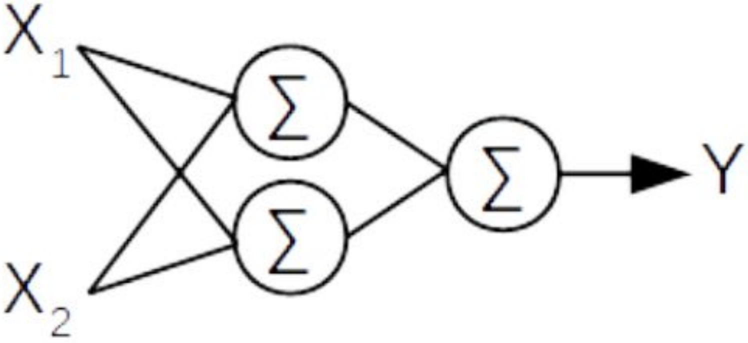
- Como resolver problemas que não são linearmente separáveis?
- Aumentar o número de neurônios resolve? Não!

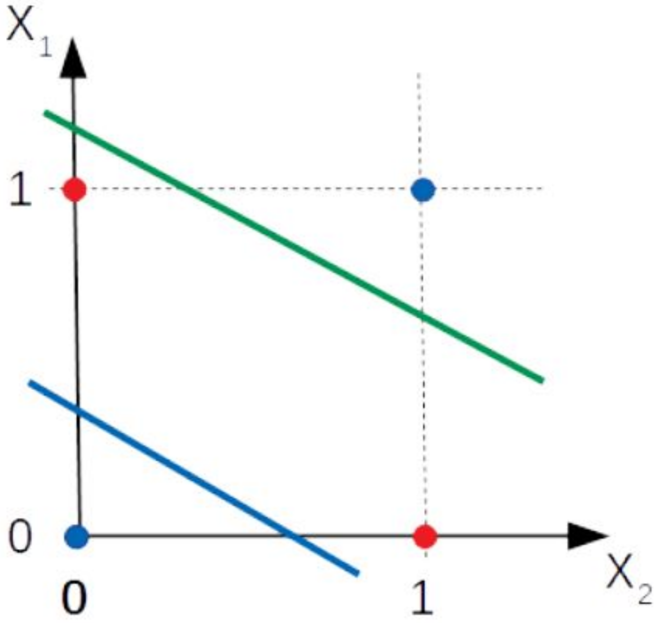




problemas e soluções

A solução é aumentar o número de camadas de neurônios.

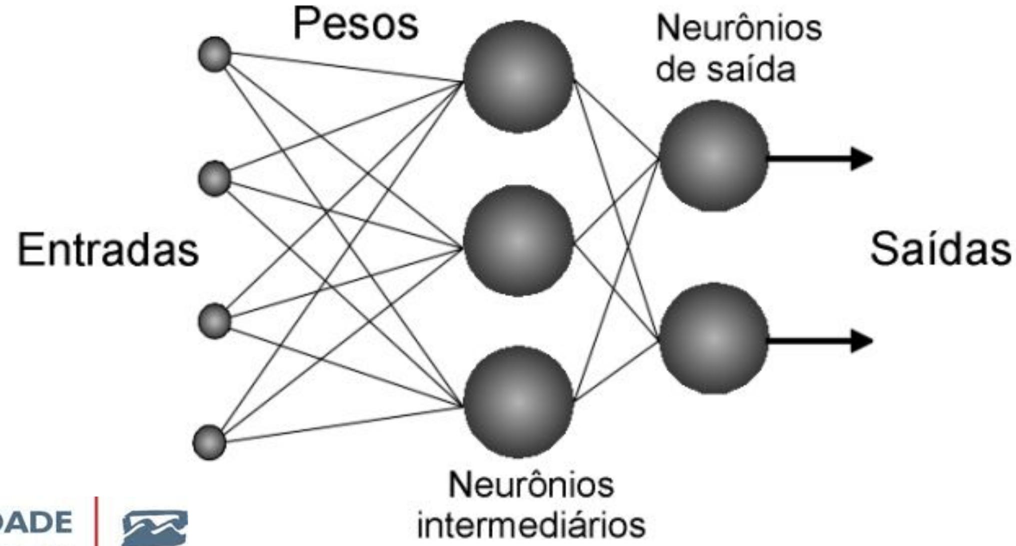




UNIVERSIDADE
DO ESTADO DE MINAS GERAIS
UEMG
UNIDADE DIVINÓPOLIS

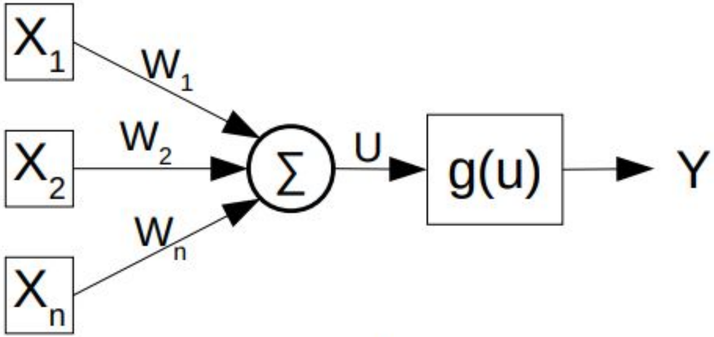
problemas e soluções

As redes neurais mais modernas são compostas por múltiplas camadas de neurônios e possibilitam a resolução de problemas não linearmente separáveis.



problemas e soluções

Para o modelo de neurônio a seguir temos:

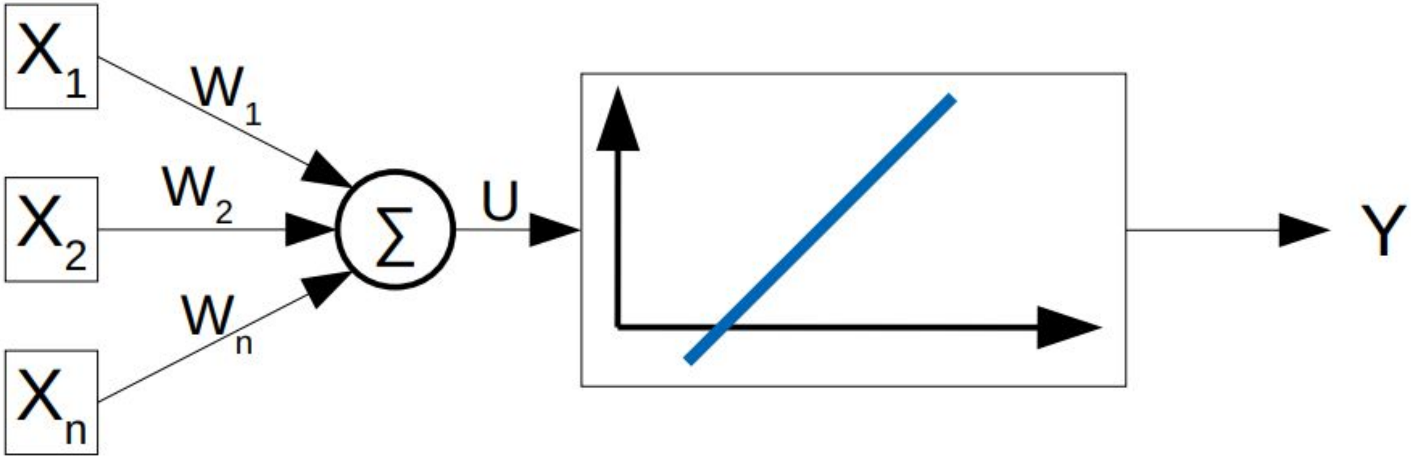


- Potencial de ativação: $u = \sum_{i-1}^{n} W_i \cdot X_i$
- Função de ativação: g( ). (função degrau nos exemplos anteriores)
- Assim: $Y = g(u) = g\left(\sum_{i-1}^{n} W_i \cdot X_i \right)$

UNIVERSIDADE
DO ESTADO DE MINAS GERAIS
UEMG
UNIDADE DIVINÓPOLIS

funções de ativação

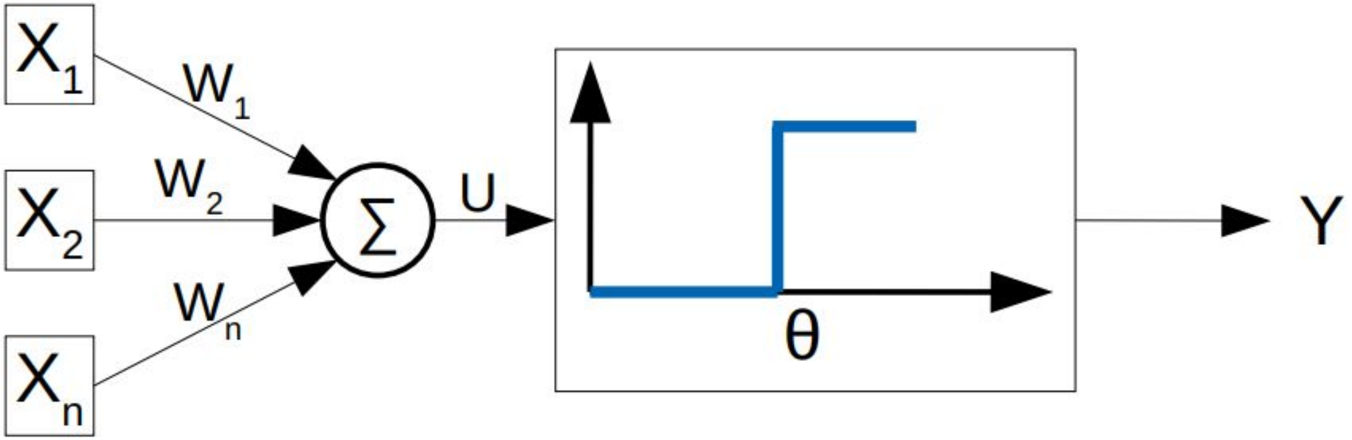
Função Linear



$$Y=\gamma\left(\sum_{i-1}^{n} W_i \cdot X_i\right)=\gamma u$$

funções de ativação

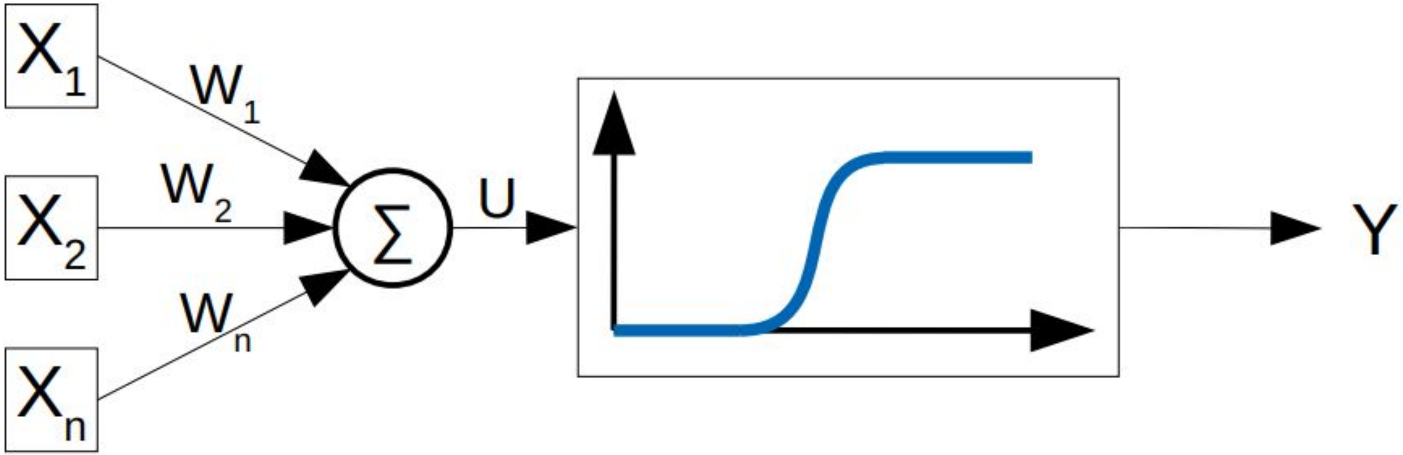
Função Degrau



$$Y = \begin{cases} 1 & se\left(u \geq \theta\right) \\ 0 & se\left(u < \theta\right) \end{cases} \quad \text{onde,} \quad u = \left(\sum_{i-1}^{n} W_i \cdot X_i \right)$$

funções de ativação

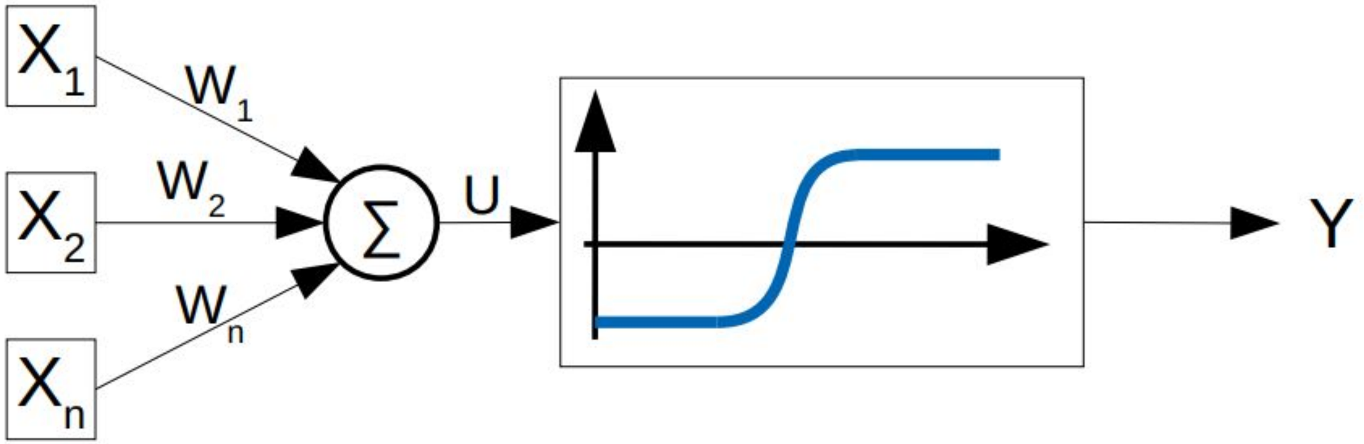
Função Sigmoidal



$$Y = \frac{1}{1+e^{-u}} \qquad onde, u = \left(\sum_{i-1}^{n} W_i \cdot X_i \right)$$

funcões de ativacão

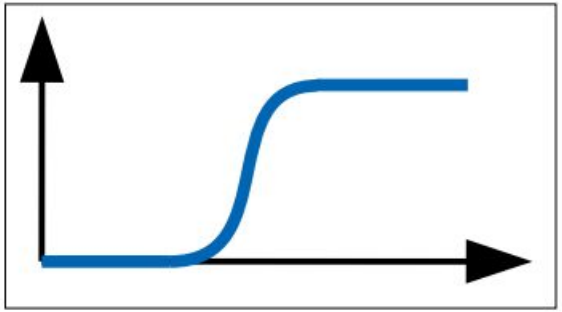
Função Tangente Hiperbólica

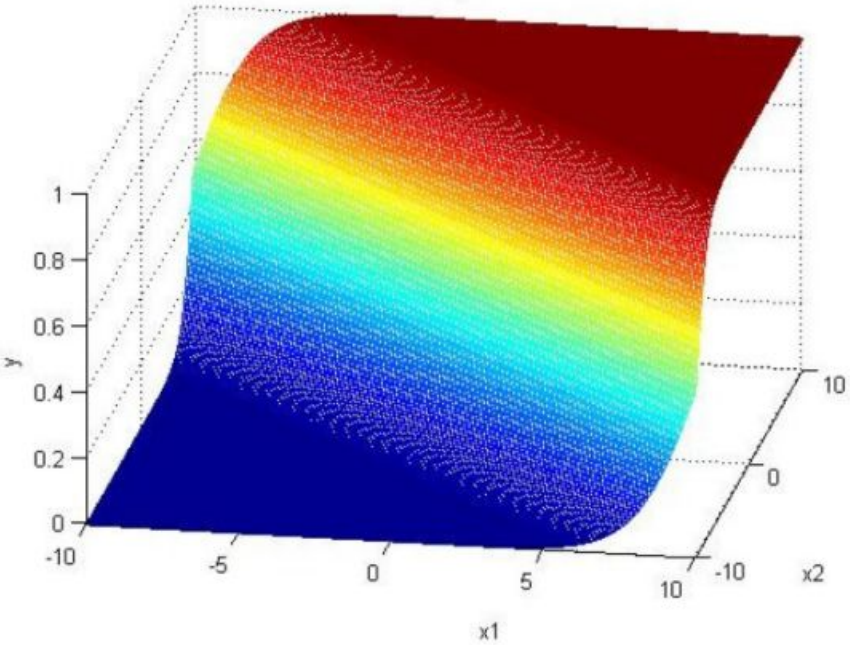


$$Y = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \quad onde, u = \left(\sum_{i-1}^{n} W_i \cdot X_i \right)$$

funções de ativação

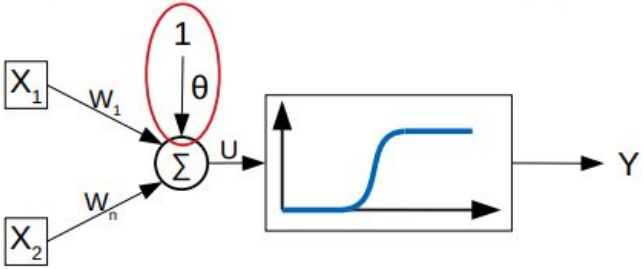
Mapeamento de entrada para saída em um neurônio com função de ativação sigmoidal.



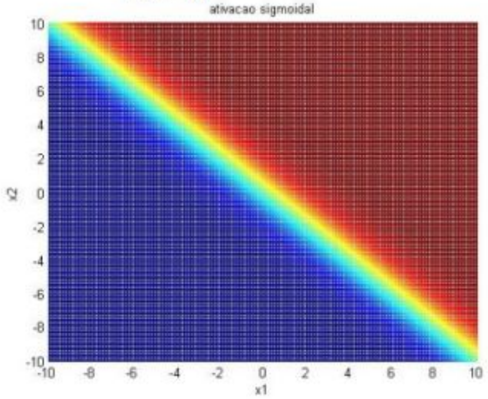


funções de ativação

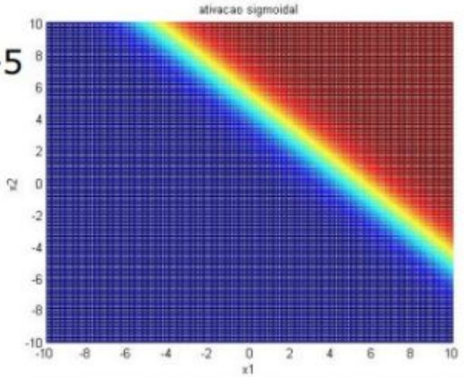
Uma entrada especial chamada **Bias** é utilizada para ajustar o limiar de atuação.



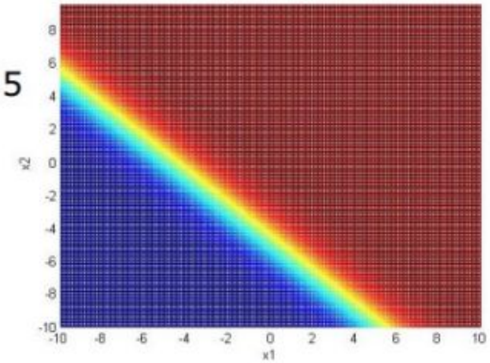
Θ =0



Θ =-5



Θ =+5

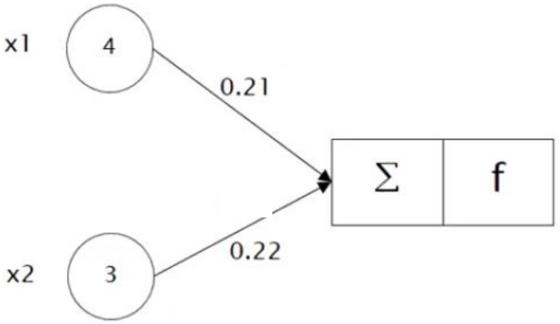


ajuste de pesos

- **classificação**

x1 - comprimento do parafuso
x2 - diâmetro do parafuso
Classe A (0) e Classe B (1)

$$soma = \sum_{i=0}^{n} xi \, * wi$$

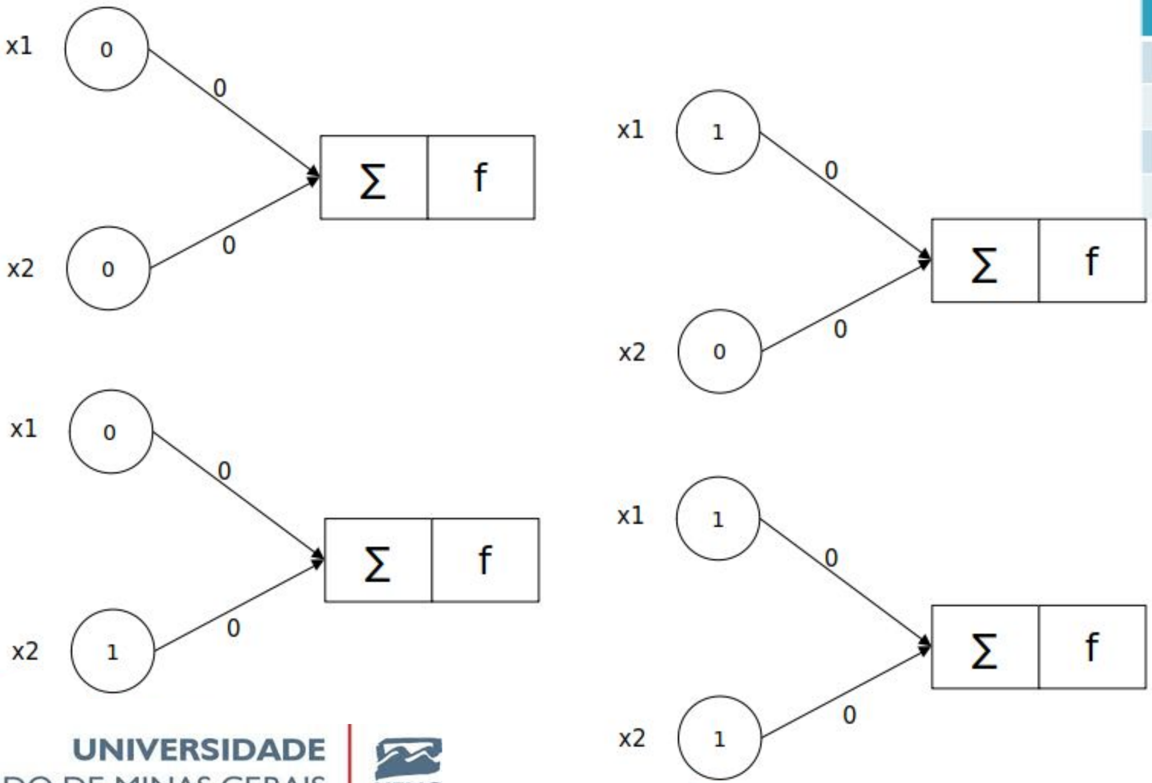


ajuste de pesos

Operador E

x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

ajuste de pesos



x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

ajuste de pesos

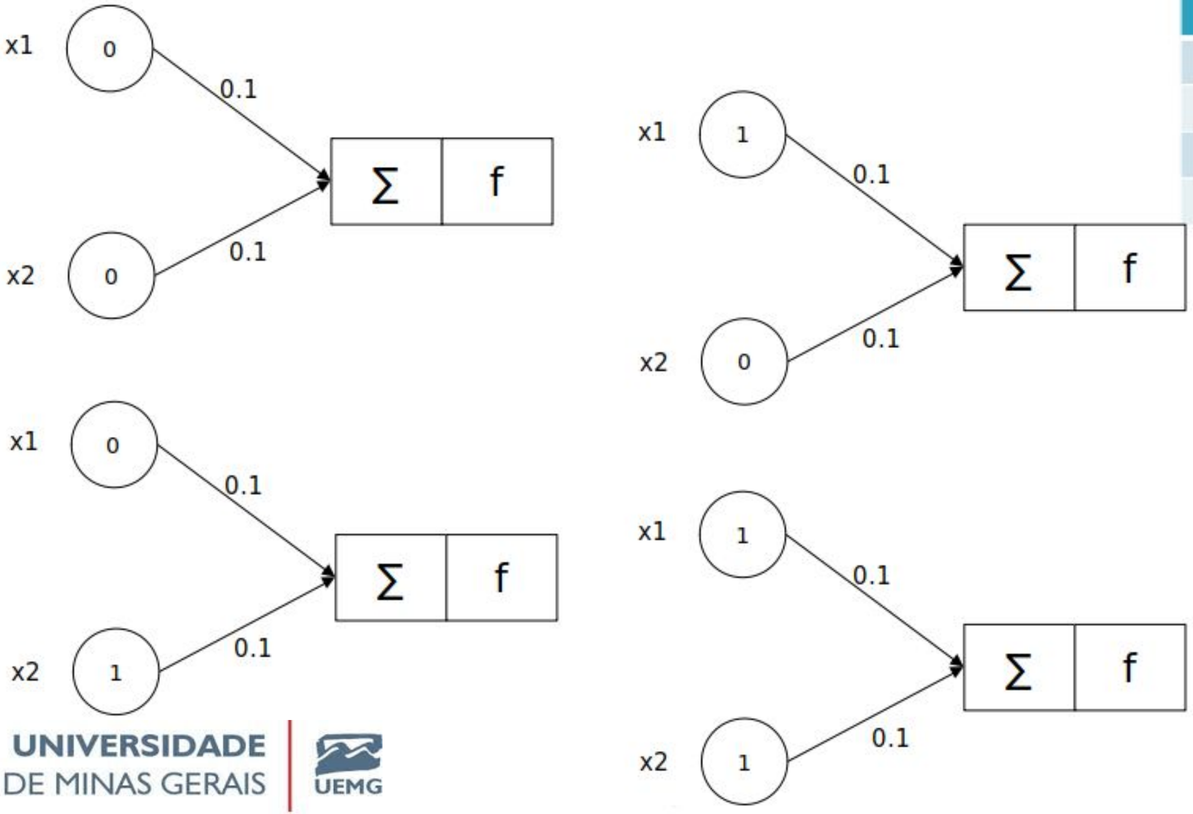
Algoritmo mais simples

- **erro = respostaCorreta – respostaCalculada**

Os pesos são atualizados até os erros serem pequenos

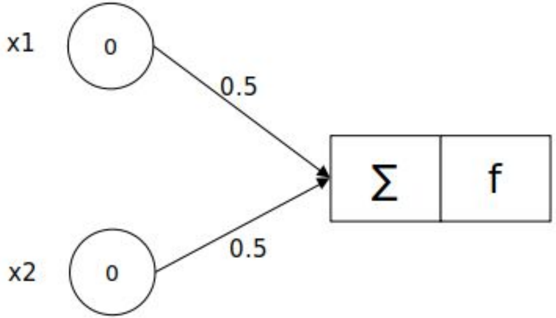
- **peso(n + 1) = peso(n) + (taxaAprendizagem * entrada * erro)**

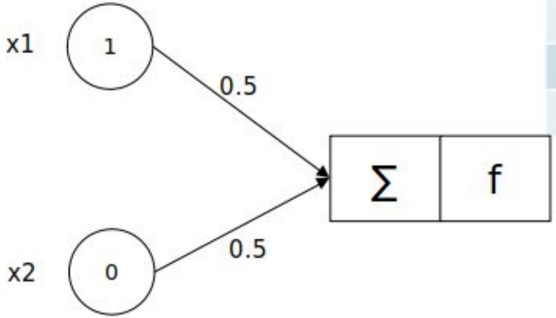
ajuste de pesos



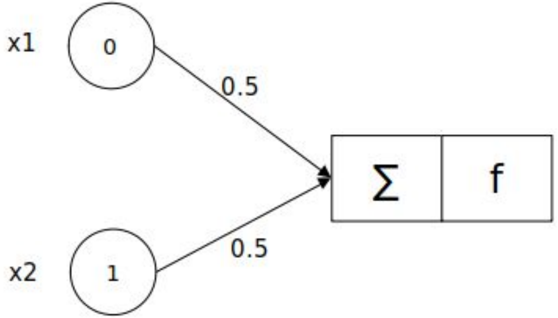
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

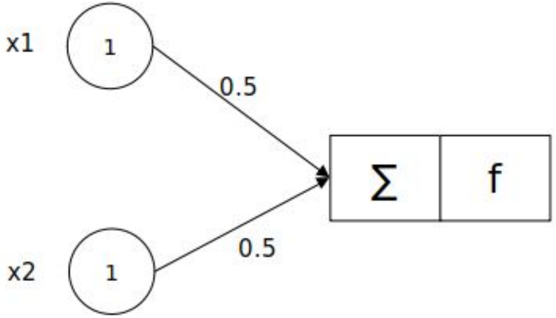
ajuste de pesos





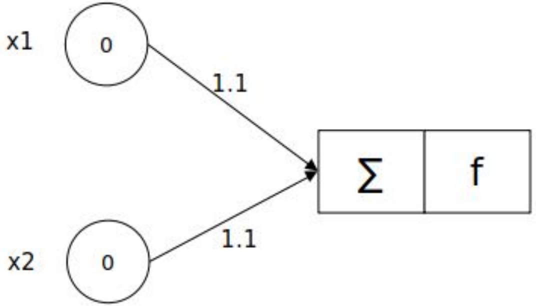
x1	x2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

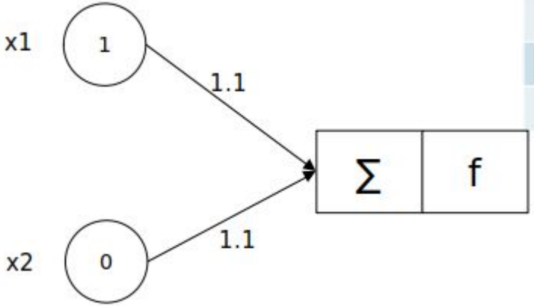


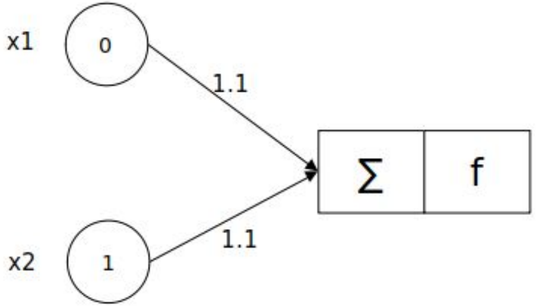


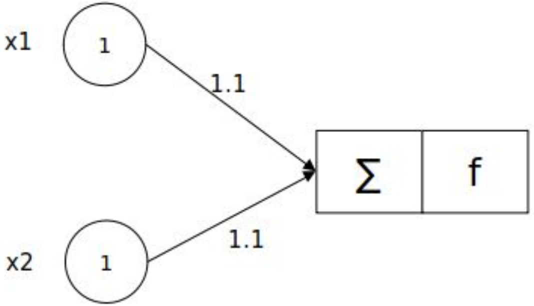
ajuste de pesos

x1	x2	Classe
0	0	0
0	1	1
1	0	1
1	1	1

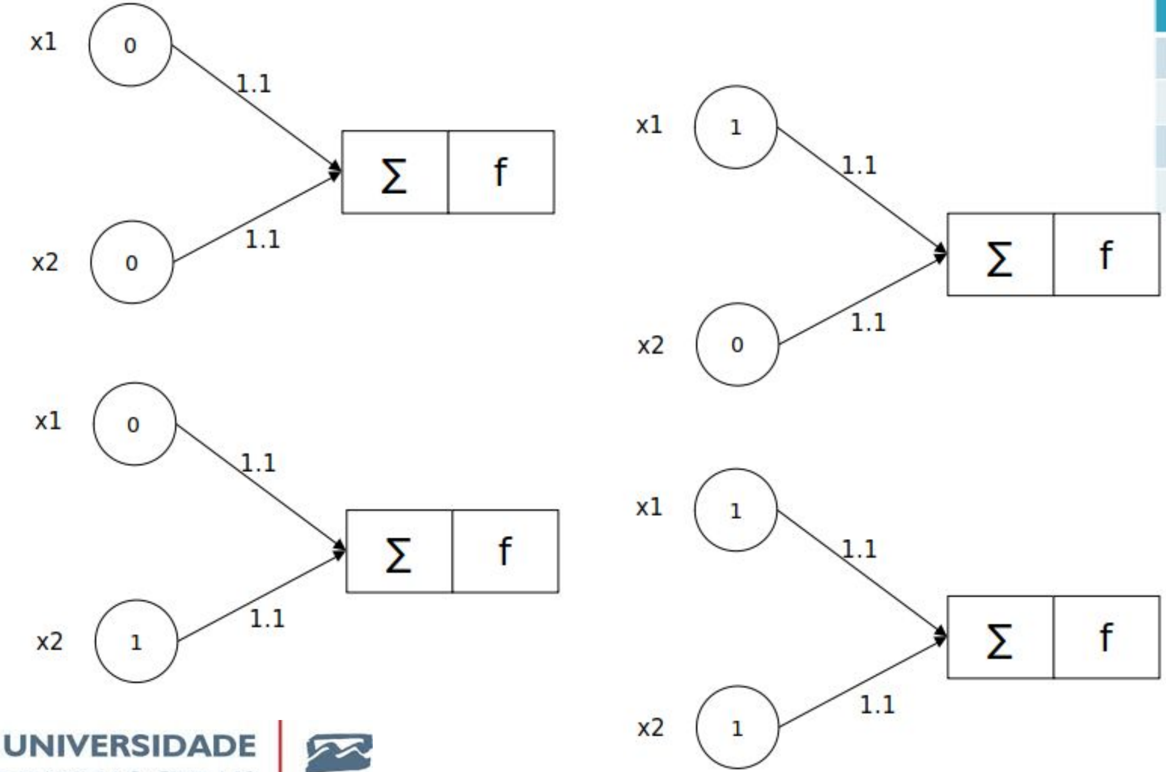








ajuste de pesos



x1	x2	Classe
0	0	0
0	1	1
1	0	1
1	1	1

ajuste de pesos

Enquanto o erro for diferente de zero

- **Para cada registro**
    - **Calcula a saída com os pesos atuais**
    - **Compara a saída esperada com a saída calculada, somando o erro**
    - **Para cada peso da rede**
        - **Atualiza o peso - peso(n + 1) = peso(n) + (taxaAprendizagem * entrada * erro)**

ajuste de pesos